R2: Boosting Liquidity in PCNs
with Online Admission Control

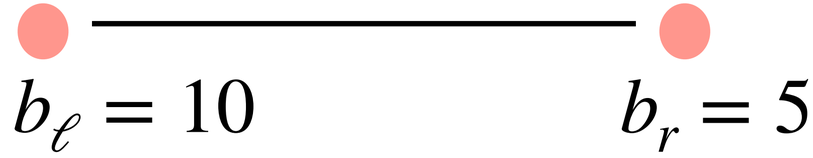
Mahsa Bastankhah[1], Krishnendu Chatterjee[2], Mohammad Ali Maddah-Ali[3], Stefan Schmid[4], Jakub Svoboda[2], Michelle Yeo[2]

[1]Sharif University of Technology [2]ISTA [3]University of Minnesota Twin Cities [4]TU Berlin

Payment Channel Networks

- Payment Channel Networks (PCNs) improve scalability of blockchains

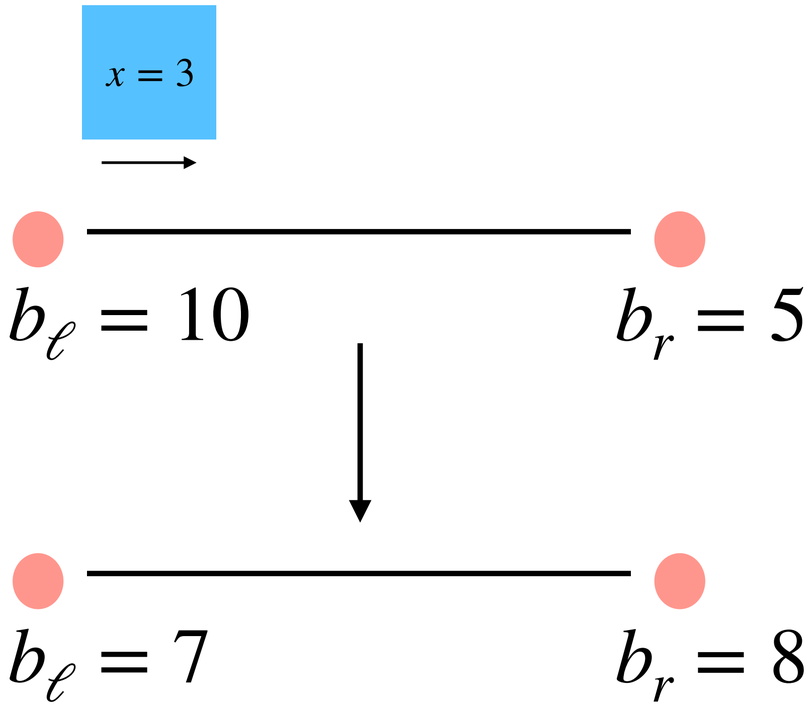
Initialisation:



Blockchain

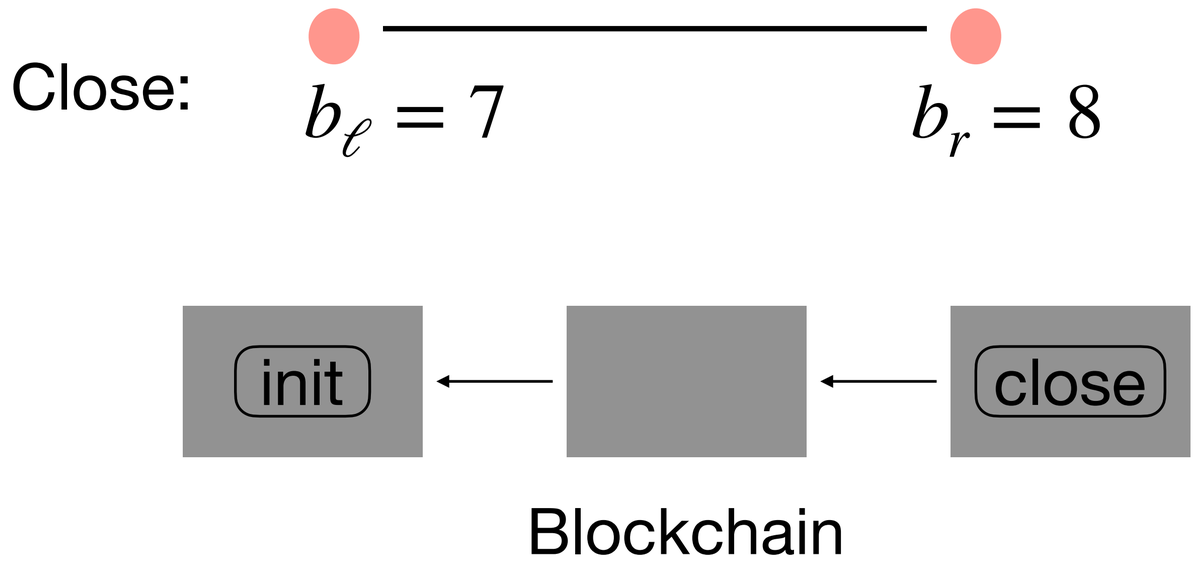
Payment Channel Networks

- Payment Channel Networks (PCNs) improve scalability of blockchains



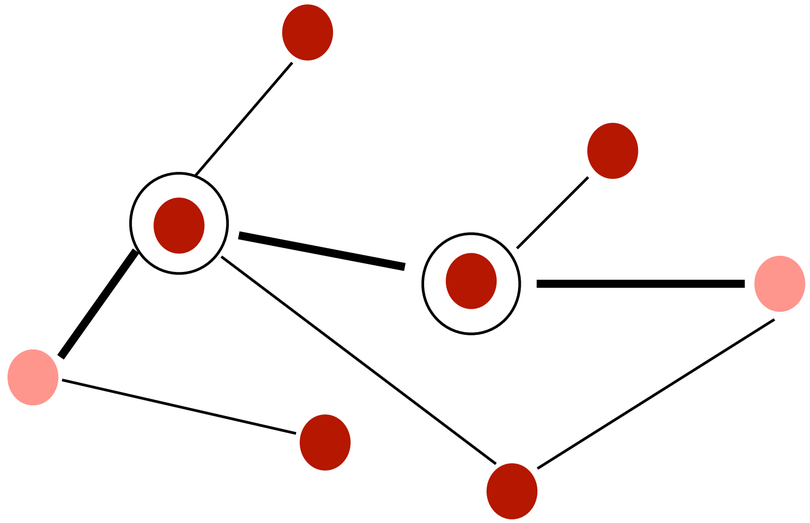
Payment Channel Networks

- Payment Channel Networks (PCNs) improve scalability of blockchains



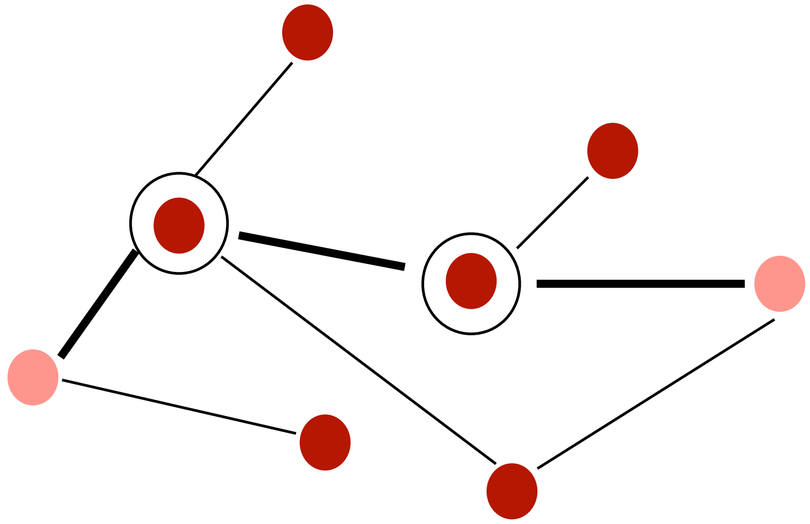
Payment Channel Networks

- Payment Channel Networks (PCNs) improve scalability of blockchains
- Intermediary nodes charge fees to forward payments
- Need to balance greedily forwarding payments and channel depletion



Payment Channel Networks

- Payment Channel Networks (PCNs) improve scalability of blockchains
- Intermediary nodes charge fees to forward payments
- Need to balance greedily forwarding payments and channel depletion



Can we design efficient algorithms to handle transactions as well as channel depletion?

Related Work

Transaction Handling

Online payment network design, Avarikioti et al. 2019

Channel depletion

Building stable off-chain payment networks, Fazli et al. 2021

Related Work

Transaction Handling

Online payment network design, Avarikioti et al. 2019

Channel depletion

Building stable off-chain payment networks, Fazli et al. 2021

This work: transaction handling and channel depletion

Problem setting

- Online decision making
- Input: transaction sequence $X_t = \{x_1, \ldots, x_t\}, x_i \in \mathbb{R}^+$
- **sequential** access to the transaction sequence

Action Space

- Actions:
  1. accept transaction
  2. reject transaction
  3. recharge channel
  4. rebalance channel

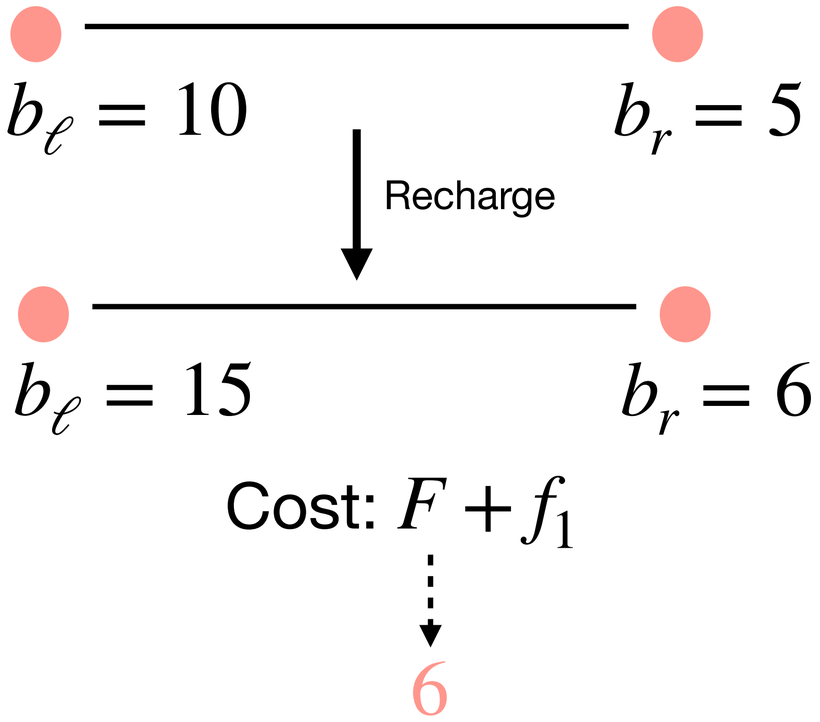
Action Space

- Actions:
    1. accept transaction
    2. reject transaction
    3. recharge channel
    4. rebalance channel

Cost: $Rx + f_2$

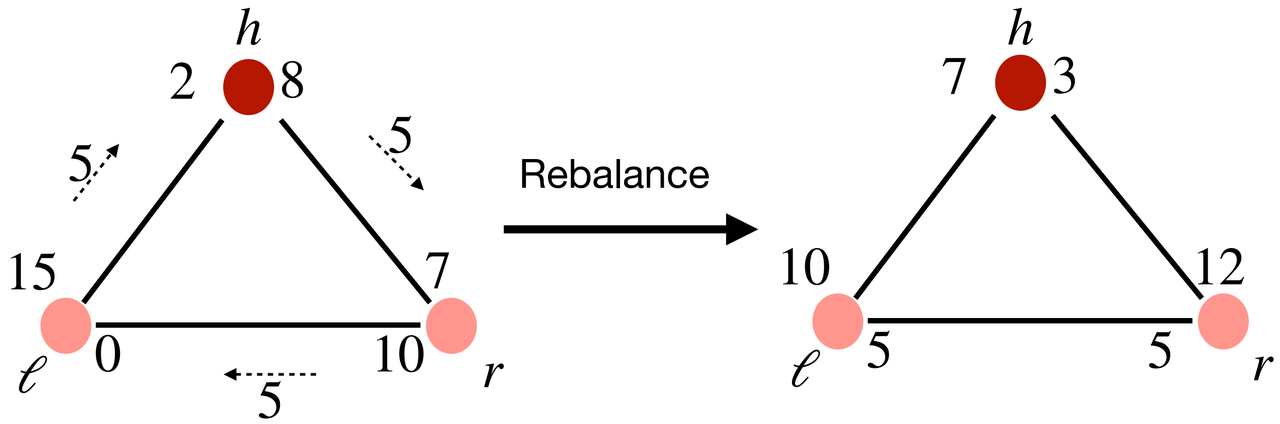
Action Space

- Actions:
    1. accept transaction
    2. reject transaction
    3. recharge channel
    4. rebalance channel



Action Space

- Actions:
  1. accept transaction
  2. reject transaction
  3. recharge channel
  4. rebalance channel



Cost: $C(Rx + f_2)$

Competitive Ratio

- Optimal offline solution (OFF): access to **entire** transaction sequence in advance
- Competitive ratio: online algorithm (ON) is c-competitive compared to OFF if for every transaction sequence X_t,

$$COST_{ON}(X_t) \leq c \cdot COST_{OFF}(X_t)$$

- Deterministic algorithms

Warm Up 1

- Unidirectional transaction stream (wlog left to right)
- ℓ can only **accept** transactions, and **recharge** channel when channel is depleted

Warm Up 1

- Unidirectional transaction stream (wlog left to right)
- ℓ can only **accept** transactions, and **recharge** channel when channel is depleted
- Real-life example: router that wants to maintain reputation as a major hub

Warm Up 1

- Unidirectional transaction stream (wlog left to right)
- ℓ can only **accept** transactions, and **recharge** channel when channel is depleted
- Real-life example: router that wants to maintain reputation as a major hub

Theorem 1

There exists an optimal 2-competitive online algorithm

Warm Up 1

- Observation: OFF knows entire sequence so will recharge only once at the start

$$COST_{OFF}(X_t) = \sum_{i=1}^{t} x_i + f_1$$

- Assume ON has access to an oracle Funds(X_i) that returns the amount of funds OFF uses in order to process sequence X_i[1]

- Tracking algorithm sequentially processes X_t calls Funds(X_i) for each prefix $X_i \subset X_t$, recharges when Funds(X_i) goes above some threshold

[1]In general this is NP-hard, but can be efficiently approximated to constant factor, see https://arxiv.org/pdf/2204.13459.pdf

Warm Up 2

- Unidirectional transaction stream (wlog left to right)
- ℓ can only **accept** and **reject** transactions, and **recharge** channel when channel is depleted

Warm Up 2

- Unidirectional transaction stream (wlog left to right)
- ℓ can only **accept** and **reject** transactions, and **recharge** channel when channel is depleted

Theorem 2

There exists a $\left(2 + \frac{\sqrt{5}-1}{2}\right)$-competitive online algorithm

Warm Up 2

Big: $x > Rx + f_2$

Small: $x \leq Rx + f_2$

Warm Up 2

- Observations:

    1. OFF rejects big transactions

    2. If there are sufficiently many small transactions to offset the cost of recharging, OFF performs a single recharging action at the beginning of size

$$\sum_{x \in X_t, x \text{ small}} x$$

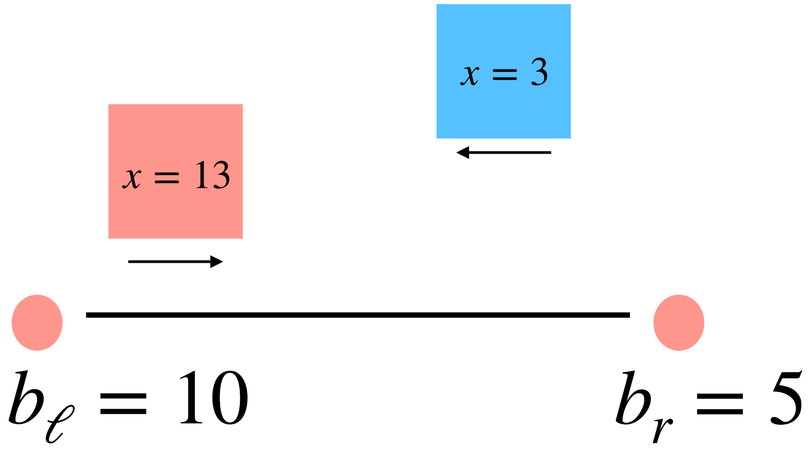
- ON similarly rejects big transactions, performs tracking to recharge channel

Big: $x > Rx + f_2$

Small: $x \leq Rx + f_2$

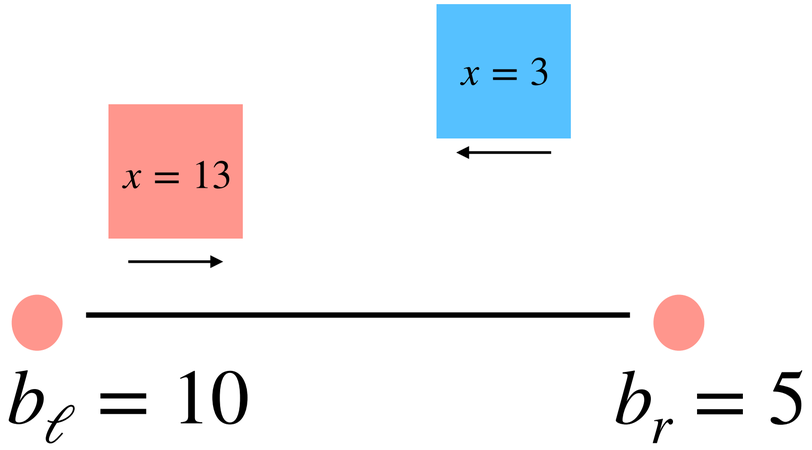
Main Problem

- Bidirectional transaction stream
- ℓ and r can **accept** and **reject** transactions, and **recharge** or **rebalance** their channel when the channel is depleted
- Want to minimise the cost over the **entire** channel (cost of **both** users)
- Real-life example: 2 routers collaborating to minimise total cost over their channel



Main Problem

- Bidirectional transaction stream
- ℓ and r can **accept** and **reject** transactions, and **recharge** or **rebalance** their channel when the channel is depleted
- Want to minimise the cost over the **entire** channel (cost of **both** users)
- Real-life example: 2 routers collaborating to minimise total cost over their channel

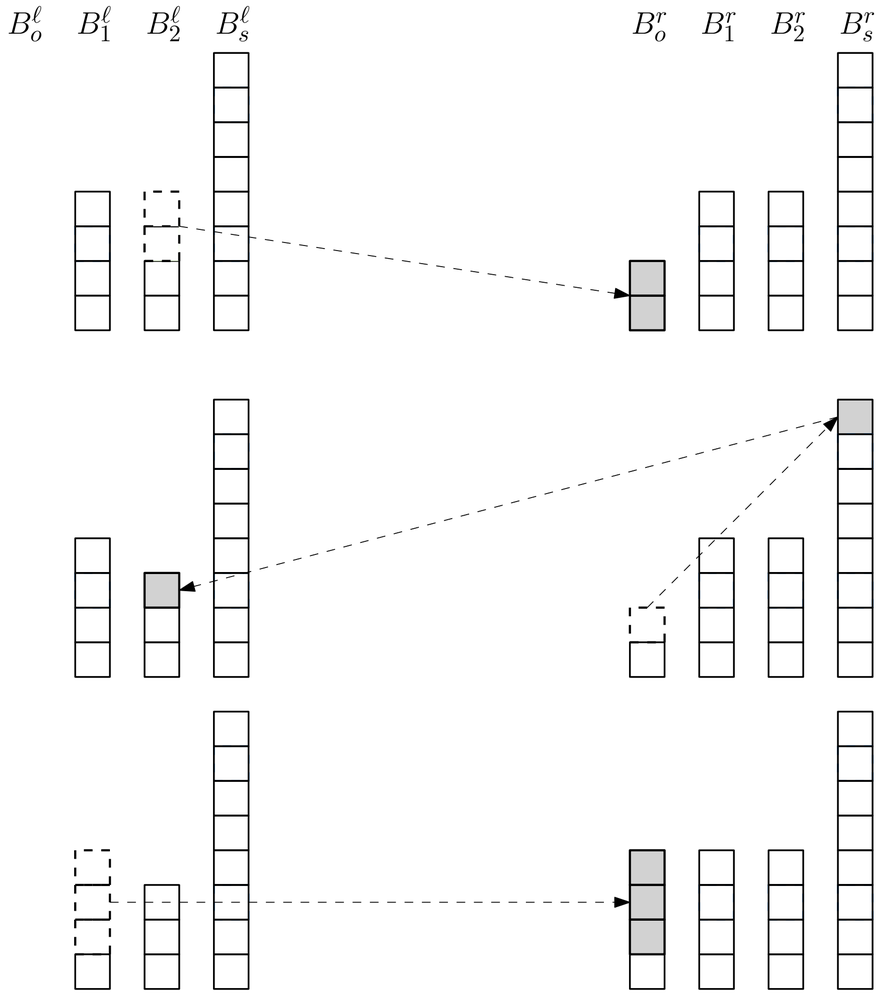


Theorem 3

There exists a $(7 + 2\lceil \log C \rceil)$-competitive online algorithm

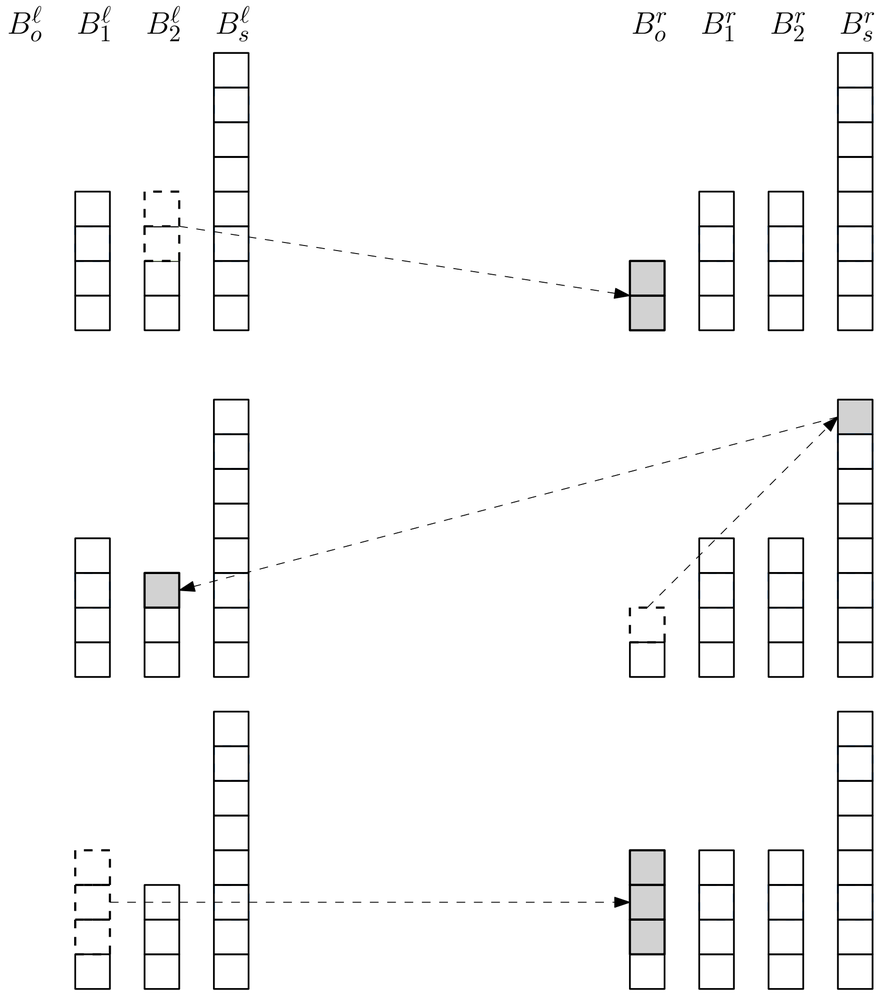
Main Algorithm

- Similar tracking algorithm to track funds of OFF in the entire channel
  - Recharges the channel whenever Funds(X_i) goes above a threshold



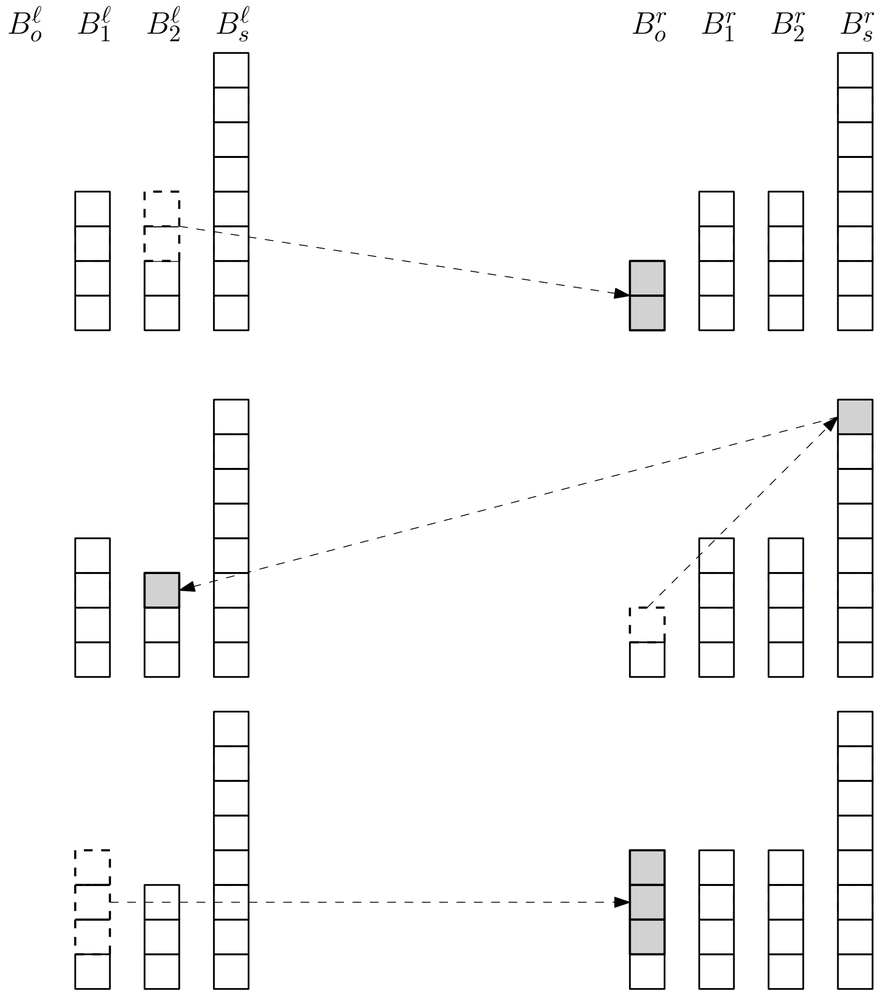
Main Algorithm

- Funds on each end of the channel divided into buckets
  - Each fund bucket is used to handle transactions of a certain size
  - If funds in a given bucket is insufficient, transaction is either rejected or off-chain rebalancing is performed



Main Algorithm

- Incoming funds are distributed to prioritise handling smaller transaction sizes
  - Rejecting smaller-sized transactions costs more



Simulations

- Compare average performance of main online algorithm on randomly generated transaction sequences (50 sequences of length 50) to our theoretical worst case bounds

Params		**OFF**	**ON**	
C	f_2	Cost	Cost	$(7+2\lceil\log(C)\rceil)OFF$
2	0.5	15.02	**63.3**	135.18
8	0.5	15.21	**87.79**	197.73
2	2	23.6	**127.02**	212.4
8	2	24.5	**184.32**	318.5

Conclusion and Future Directions

- Study the problem of online transaction handling and channel depletion in the general setting
- Provide algorithms with worst-case guarantees
- Simulations show significantly better average case performance
- Future work:
  - Consider randomised algorithms
  - Incorporate fees into the decision problem

Questions?
myeo@ist.ac.at